Unsteady Flow Volumes

Barry G. Becker*
David A. Lane**
Nelson L. Max*

*Lawrence Livermore National Laboratory
P.O. Box 808 / L-301
Livermore, CA 94551
(becker1@llnl.gov, max2@llnl.gov)

**Computer Sciences Corporation
NASA Ames Research Center
M/S T27A-2
Moffett Field, CA 94035
(lane@nas.nasa.gov)

Abstract

Flow volumes [1] are extended for use in unsteady (time-dependent) flows. The resulting unsteady flow volumes are the 3 dimensional analog of streaklines. There are few examples where methods other than particle tracing have been used to visualize time varying flows. Since particle paths can become convoluted in time there are additional considerations to be made when extending any visualization technique to unsteady flows. We will present some solutions to the problems which occur in subdivision, rendering, and system design. We will apply the unsteady flow volumes to a variety of field types including moving multi-zoned curvilinear grids.

Introduction

Visualization of steady flow is often based on streamlines. A streamline is a curve which is everywhere tangent to a vector field at a moment in time. For unsteady flows one may use pathlines, timelines or streaklines as a basis for dynamic visualization techniques. A pathline shows the trajectory of a particle released from a given location. Timelines are rakes of connected points that have all been released at the same time. Streaklines are the curves formed by joining the positions, at an instant in time, of all the particles which have been released previously from a single location. The unsteady flow volumes presented in this paper follow the analogy of streaklines.

There are a variety of standard methods used to visualize flow fields, many of which can be extended to unsteady flows in a straightforward manner. Hedgehogs, contour surfaces of constant velocity magnitude, and particle traces are good examples because they consist of sets of independent graphics primitives.

Streamlines, ribbons, and volumes provide increasingly informative representations of a flow. None of them, however, is trivial to extend to unsteady fields because of the dynamic relationship between vertices. There is no guarantee that consecutive vertices on the same streakline will remain within a given distance apart from one time step to another. In static flow fields, the vertices do not move so this problem does not arise.

Multi-zoned curvilinear grids are commonly used in computational fluid dynamics (CFD) simulations. These grids may have rigid body motions. Lane's [2] particle tracing package provides for advection in these types of flows. We will make use of his UFAT library to advect the vertices of the unsteady flow volume when in a multi-zoned or moving curvilinear grid.

There are many methods which give good representations of static fields but have never been applied to unsteady flows or non-regular grids. Stream surfaces [3], cloud tracing [4], virtual smoke[5], surface particles [6], and spot noise [7] are examples of these. Spot noise may be a successful technique for unsteady flows because it does not try to maintain connected primitives.

Techniques which use advection may benefit from calling UFAT [2] if they are to be applied to multi-zoned curvilinear grids which may contain moving grids. For methods like texture splats [8], which do not rely on particle advection, unusual grids must be handled in another way.

There are few examples where visualization other than particle tracing has been successfully applied to un-

steady flows. Crawfis and Max provide an example by advecting cloud textures on contours of percent cloudiness in [9]. Unfortunately their method required looping through every time step for each frame of animation. Even streaklines are not common because there is a tendency for adjacent particles to become too far apart. This implies that smooth lines may quickly become jagged.

Steady Flow Volumes

A flow volume is the 3D equivalent of a streamline, and consists of the union of the streamlines that start on a 2D generating polygon [1]. This volume is divided up into a set of semitransparent tetrahedra, which are volume rendered in hardware in a way derived from the method of Shirley and Tuchmann [10]. The tetrahedra are arranged to triangulate the slabs between the layers of vertices in successive time steps. Therefore construction in a steady flow entails adding slabs of extra tetrahedra to the leading surface of the volume for each new time step (see figure 1). A curvature based method of adaptive subdivision was used to increase resolution if the flow diverged too much.

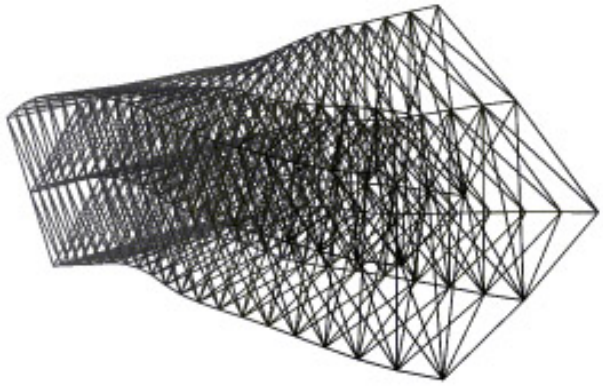

Figure 1

If the flow volume is monochrome no sorting is necessary. Often it is desirable for each vertex in the flow volume to have color and opacity assigned to it according to an independent scalar quantity. In this case the sorting and hardware assisted rendering described by Stein [11] is used. The smoke may be combined with solid geometry since opaque polygons are rendered into the z-buffer first.

There is an option to divide the opacity by the tetrahedron's volume. This will make regions where the smoke is compressed more opaque because the relative volume will be smaller.

Instead of shortening the time step (*dt*) until a desired accuracy is reached, it is better to use an improved integration scheme. This will reduce the number of graphical primitives needed to represent the same amount of accuracy. Besides Euler and Runge-Kutta (RK) methods, adaptive methods try to modify each successive integration step so that the error is kept constant. The method described in Press *et al* [12] either lengthens or shortens the time step to accomplish this. Figure 2 shows how this can improve results for a 4th order RK scheme. Let $\mathbf{P}_n$ be the current position at t_n and $\mathbf{P}_{n+1}$ be the next position at t_{n+1}. The following pseudo code finds the next position $\mathbf{P}_{n+1}$ and calculates the *dt* to be used in the next time increment t_{n+1}, that will keep the error within some desired tolerance ε.

$\mathbf{V}$= velocity($\mathbf{P}_n$)
$\mathbf{P}_{mid}$ = RungeKutta($\mathbf{P}_n$, 0.5 *dt*)
$\mathbf{P}_{full}$ = RungeKutta($\mathbf{P}_n$, *dt*)
$\mathbf{V}_{mid}$ = velocity($\mathbf{P}_{mid}$)
$\mathbf{P}_{n+1}$ = RungeKutta($\mathbf{P}_{mid}$, 0.5 *dt*, $\mathbf{V}_{mid}$)
error = $|\mathbf{P}_{n+1} - \mathbf{P}_{full}|$
if (error > ε)
    dt =0.9 *dt* ($|\varepsilon$ / error|) ^{SHRINK}
else
    dt = 0.9 *dt* ($|\varepsilon$ / error |) ^{GROW}

where SHRINK is 1/2 or 1/4 for a second or fourth order RK scheme respectively, and GROW is 1/3 or 1/5. The derivation of these exponents is described in [12]. This adaptive scheme requires 11 rather than the 8 function evaluations used if we did just the two half steps. The factor of 0.9 keeps the change in *dt* conservative.

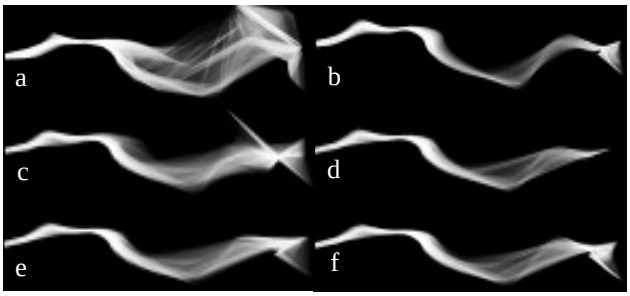

Figure 2. a) Euler, b) exact, c) 2nd order RK, d) adaptive 2nd order RK, e) 4th order RK, f) adaptive 4th order RK.

Unsteady Flow Volumes

Flow volumes for unsteady flows correspond to the union of streaklines that start on the generating polygon. The volume evolves in time by moving "sideways" as the streaklines shift with the varying flow, as well as by adding tetrahedra at the generating polygon. This makes adaptive subdivision far more difficult, as does the use of adaptive time steps.

Adaptive Subdivision

Unsteady flow fields present some unique problems for visualizations with connected graphical primitives like lines, ribbons, and volumes. The adaptive time integration pseudo-code shown in the previous section can no longer be applied because the whole flow constructed at each time step needs to be advected, not just the last particle or layer of particles.

Using the adaptive time integration just described for advecting particles in unsteady flows would generate intermediate particle positions between the given time steps. However, for visualization, the particle positions are displayed only at the given time steps. Hence, the adaptive stepping described by Lane in [2] is used.

Another problem is the difficulty in extending the old subdivision scheme of [1] or [13] to unsteady flows. Unlike static flow volumes it is possible for adaptive subdivision to be required at any point along the existing flow rather than at the flow front only. Also a provision for reversing the subdivision when it is no longer needed should be included. The reason for this is that an area of the smoke flow which needs subdivision at one time step may not need it in subsequent ones. The static subdivision method provides no way to subdivide the middle region without adding new particles to the flow front. Those new particles, if they are added, must come about by advection from the first time step. It is now much more expensive to maintain a list of midpoints over every segment in the whole flow volume, as was done in [1] to decide when subdivision was necessary.

There are several ways of performing adaptive subdivision. For static flow volumes, a test based on curvature or edge length was used. This subdivision occurred only at the flow front as advection progressed, not between layers corresponding to adjacent time steps. Subdivision never went beyond a maximum amount to prevent an exponential increase in the number of tetrahedra. Once that maximum was reached it was never reduced. In turbulent flows it was common for the subdivision to reach its maximum right away, and also to miss features right at the base since the number of tetrahedra would never more than double between successive layers. Hence, for unsteady and some steady flows the amount of subdivision should be fixed at the maximum from the start.

Besides cross-sectional subdivision there is also subdivision in the direction of time to be considered. The space between layers of particles should stay approximately constant. In the steady state case we handled this using an adaptive time step. Since this is no longer possible, the solution we propose is to adaptively add or delete intermediate layers of vertices as necessary. If the average distance among all corresponding points in adjacent levels is above some tolerance, then we will insert a whole layer of vertices at the midpoints between corresponding vertices of the last two layers.

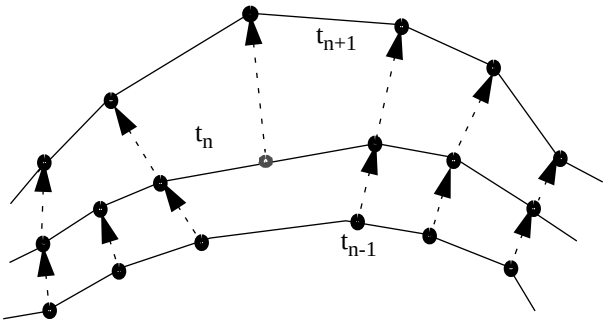

Figure 3. Adaptive insertion and deletion of particles on a streakline.

There will now be two slabs where there was only one, and the opacity of each new tetrahedral element will be half the ones they replace. These new points will be advected from this point on. They do not lie on the true streakline, since they were not advected from the first time step, but they will be a far better approximation than never letting the midpoint be advected. Similarly, if the average distance between all paired points on adjacent levels is below some tolerance then we delete that layer of points and replace the two levels of tetrahedra with one having double the opacity. The deleting of vertices may cause problems later if the remaining neighbors start to spread apart again. Perhaps vertices should never be deleted since it is impossible to determine the future behavior of the field. Adding particles cannot decrease the accuracy, but it can lead to prohibitively many vertices if deletion is not allowed.

In an approach described by Shariff *et. al.* [14] whenever resolution is lost due to stretching, time is returned to the instant when the group of vertices were inserted and new interpolated vertices are added. This is a time consuming, but completely accurate process. A similar idea would be to do a complete trial run to determine in a preprocess step where subdivision will be necessary. Then in the actual run particles can be added from the beginning using the hindsight provided by the trial run. This assumes a batch mode of operation which is not always desirable.

Figure 3 illustrates our method applied to a section of a streakline. A vertex is inserted at the midpoint of a segment which has exceeded the tolerance at time t_n. A vertex is deleted at time t_{n+1} when two adjacent segments become too short. Figure 4 shows frames from an animation in which a streakline with adaptively added and deleted particles is compared to one using only original particles. It is expected that this adaption scheme will work best as long as the cross-sectional subdivision is not too detailed

and the generating polygon is not too large. If the variation of distances among vertex pairs in adjacent levels is too large this method will be of little benefit. This becomes increasingly likely as an unsteady flow volume matures, and adjacent streaklines become independent. At that point the importance of cross-sectional subdivision increases.

As an unsteady flow volume advances there is a strong tendency for parts of it to dissipate. In these areas where the particles become too far apart one might expect subdivision to be crucial, but in fact it is less so because the volumes are becoming larger and hence the smoke is increasingly transparent. Although the tetrahedra are becoming large, their opacity contribution is diminished. Similarly, places where vertices get crushed together will be opaque and may be important, so small tetrahedra should not necessarily be joined into larger ones.

A user of this technique should note that the accuracy of its depiction of underlying physics degenerates over time. Adaptively adding particles significantly slows this degradation however.

Thin Tetrahedra

Inadequate subdivision will lead to long thin tetrahedra which will show up prominently (see Figure 5a). The reason for this is that the opacity of a tetrahedron is inversely proportional to its volume. This problem is present in steady flows, but compounded by unsteady ones.

There are times when tetrahedra will become stretched out in one direction while staying thin in another. These long thin tetrahedra can cause artifacts because their volume can be arbitrarily small. This stretching is common when the flow splits around an object or when the tetrahedron is near the grid surface. Tetrahedra joining stream lines on either side of the object may actually pass through it. They are dealt with by multiplying the volume by

$$\frac{E_l}{(E_s \cdot tol)}$$

if $E_l/E_s > tol$. E_l is the longest edge, E_s is the shortest, and *tol* is the largest ratio permitted. This results in greatly reducing the adverse effects of the degenerate tetrahedra by reducing their opacity and hence their visibility (see Figure 5b).

Implementation

We created a suite of Explorer modules to generate smoke in a variety of vector fields. Based on the kind of mesh it receives, a Smoke module will create an instance of the appropriate type of vector field. The vertices of the flow volume are particles that maintain their positions with respect to time. There are different types of particles for each type of vector field. If the field is unsteady then the network becomes more complicated (see Figure 6) because the smoke module needs to read the next vector field before each new advection step. Changing the step size, amount of subdivision, or starting position will immediately cause the vector field to return to its first time step, hence starting the flow over with the new parameter. Changing the color or transparency will not restart the flow. The flow will not advect until the advect button is pressed. This allows the flow volume to be rotated or moving puffs with cyclically varying opacity to be enabled while the motion in the unsteady flow is paused. The "number of steps" slider marks the maximum number of allowable levels in the flow. If the number of layers exceeds this maximum, either because of natural advection or because too many layers are adaptively inserted, then the flow will be truncated.

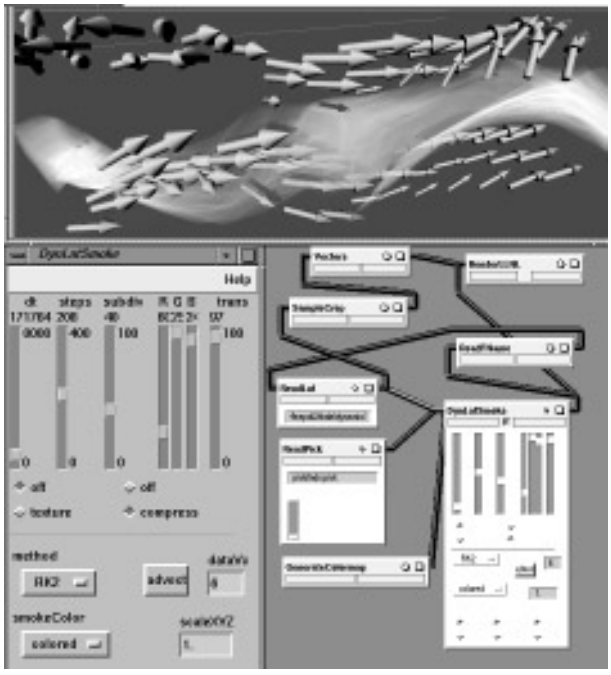

Figure 6. An Explorer map for generating unsteady flow volumes. A new vector field is read in for every new advection step.

We created a hierarchy of C++ vector field classes (see Figure 7). Through polymorphism we avoid rewriting or copying most functions. All particles share a common interface. The implementation of that interface varies greatly depending on the type of particle, but the code which uses particles (such as the routine which constructs the flow volume) only calls functions defined for all the types of particles.

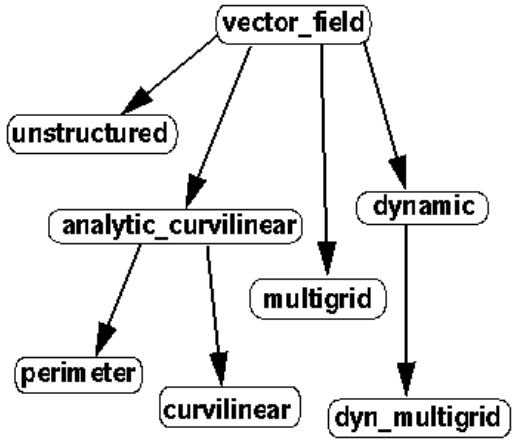

Figure 7

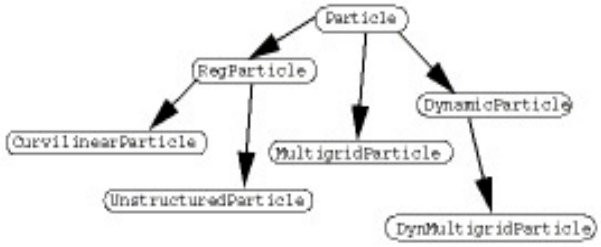

Figure 8

Multiple grid vector fields, a subclass of the base vector field, contain an array of pointers to vector fields of any type. Dynamic vector fields, also a subclass of the base vector field, contains pointers to the two vector fields which are currently loaded into memory at the current time. Those two fields may even have multiple grids. As time moves forward the last field will be deleted and the next one will be loaded in so that only fields representing the current and last time steps reside in memory at any one time. Each class of vector field has in its standard interface virtual methods for sampling values. Any class of particle can set its own physical position, copy itself, find a midpoint between itself and another particle, and advect to a new position. Figure 8 shows a structure for particles similar to the one for vector fields.

A flow can be colored by any available independent scalar variable like pressure or temperature. Coloring the flow by time released is also a feature. Particles in dynamic flows record the time they were released. This value may be used to assign colors to vertices just as any other scalar value which is present in the field. Coloring the flow also provides an important depth cue not present in monochrome smoke.

Results

We have used flow volumes to interactively explore a wide range of data sets, and have created several interesting animations. Figure 9 shows an unsteady flow volume in a curvilinear vector field of simulated wind data over Indonesia after 72 time steps (sampled over 3 days in January). The smoke is colored by its age. The animation shows much more information. Figure 10 depicts a frame from an animation showing the flow volume technique applied to an unsteady data set of a clipped delta wing with oscillating flaps. The delta wing data set is composed of four curvilinear grids consisting of a quarter of million grid points. A total of 5,000 time steps per oscillation are created in the numerical simulation. However, only every 50^{th} time step is saved. In the animation, particles were released near the surface of the wing and an unsteady flow volume is dynamically constructed. Finally Figure 11 shows an unsteady flow volume near a missile in flight. The missile is in a curvilinear grid and consists of a half million points.

After each advection step the scene may be interactively rotated. The actual advection requires reading in the next field and advecting the entire flow volume and hence is typically not interactive.

The time for constructing and displaying the flow volume at each step is proportional to the number of vertices, while reading the next vector field at each step takes time proportional to the number of sample points in the field. A major performance bottleneck is in disk access. Except for small problems, memory can hold only a few time steps simultaneously.

Images, animations, source code, and Explorer modules, are available on the Web at: http://www.llnl.gov/graphics.

Future Work

We have only presented one possible way of representing unsteady flow volumes. Better adaptive subdivision schemes should be examined. If part of the flow volume touches a region which is outside the valid domain, it is truncated at that point. A way to clip the volume without truncating should be devised. Perhaps the best thing to do is abandon meshes altogether in favor of pursuing splats.

UFAT saves the particles at each time step for playback later. Flow volumes could benefit from a similar scheme. The Explorer ReadGeom and WriteGeom modules would have to be modified to recognize flow volume geometry.

Another goal is to explore unsteady vector fields using the virtual wind tunnel created at NASA.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract number W-7405-ENG-48, with specific support from an internal “LDRD” research grant. The second author would like to acknowledge NASA for supporting portions of this work performed under contract NAS 2-12961. We wish to thank Roger Crawfis, Cliff Stein, and David Kenwright for their many suggestions, Carl Hsieh for the missile data set, John Zych for creating our Web page, and the IEEE computer Society reviewers for comments which improved the paper.

References

[1] Max, N., Becker, B. and Crawfis, R., (1993) “Flow Volumes for Interactive Vector Field Visualization”, *Proceedings of Visualization ’93*, IEEE Computer Society Press, Los Alamitos, CA, pp 19-23.

[2] Lane, D., (1994) “UFAT - a Particle Tracer for Time-Dependent Flow Fields”, *Proceedings of Visualization ’94*, IEEE Computer Society Press, Los Alamitos, CA, pp 257-264.

[3] van Wijk, J., (1993) “Implicit Stream Surfaces”, *Proceedings of Visualization ’93*, IEEE Computer Society Press, Los Alamitos CA, pp 245-252.

[4] Ma, K. and Smith, P., (1993) “Cloud Tracing in Convection Diffusion Systems”, *Proceedings of Visualization ’93*, IEEE Computer Society Press, Los Alamitos, CA, pp 253-260.

[5] Ma, K. and Smith, P., (1993) “Virtual Smoke: An Interactive 3D Flow Visualization Technique”, *Proceedings of Visualization ’92*, IEEE Computer Society Press, Los Alamitos, CA, pp 46-53.

[6] van Wijk, J., (1993) “Rendering Surface Particles”, *IEEE CG&A*, Vol. 13, No. 4, July, pp 18-24.

[7] Max, N., Crawfis, R. and Grant, C., (1994) “Visualization of 3D Vector Fields near Contour Surfaces”, *Proceedings of Visualization ’94*, IEEE Computer Society Press, Los Alamitos, CA, pp 248-255.

[8] Crawfis, R. (1992) “Texture Spats for 3D Scalar and Vector Field Visualization”, *Proceedings of Visualization ’92*, IEEE Computer Society Press, Los Alamitos, CA, pp 261-266.

[9]Crawfis, R. and Max, N., (1992) “Direct Volume Visualization of Three-Dimensional Vector Fields”, *Proceedings of the 1992 Workshop on Volume Visualization*, Kaufman and Lorensen (eds), ACM SIGGRAPH, NY, pp 55 - 60.

[10] Shirley, P. and Tuchman, A. (1990) “A Polygonal Approach to Direct Volume Rendering”, *Computer Graphics*, Vol. 24 No.5, pp 63-70.

[11] Stein, C., Becker, B., and Max, N., (1994) “Sorting and Hardware Assisted Volume Rendering”, *Symposium on Volume Rendering*, ACM Press, New York, NY, pp 83-89.

[12] Press, William, et al., (1988) *Numerical Recipes in C*, Cambridge University Press, Cambridge, pp574-578.

[13] Hultquist, J., (1992) “Constructing Stream Surfaces in Steady 3D Vector Fields”, *Proceedings of Visualization ‘92*, IEEE Computer Society Press, Los Alamitos, CA pp 171-178.

[14] Shariff, K., Pulliam, T. and Ottino, J., (1991) “A Dynamical Systems Analysis of Kinematics in the Time-Periodic Wake of a Circular Cylinder”, *Lectures in Applied Mathematics*, American Mathematics Society, Vol. 28, pp 613-646.

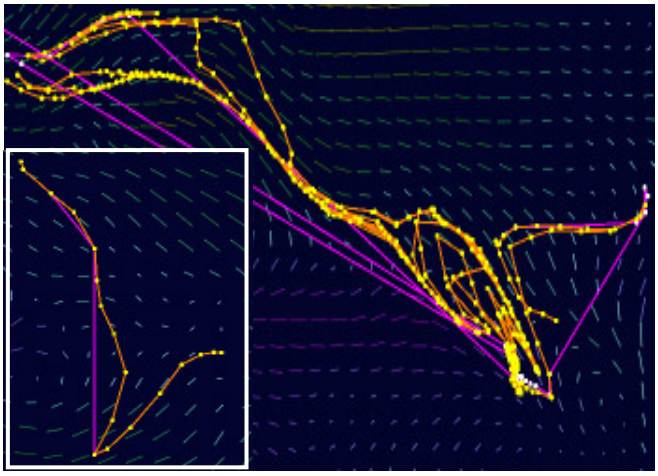

Figure 4. The orange streakline has particles adaptively added and deleted as determined by segment length. The purple streakline does not. The inset shows an earlier frame in the animation.

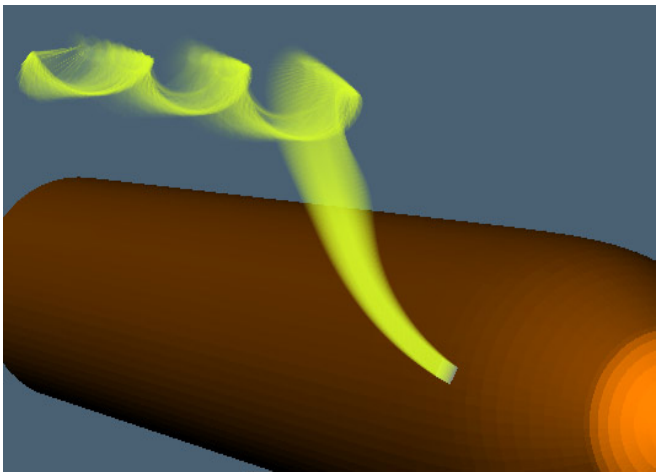

Figure 11. Unsteady flow volume near a missile in a curvilinear grid.

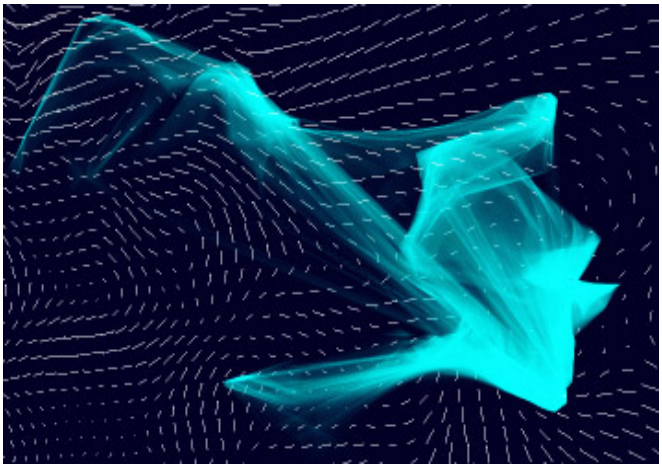

Figure 5a. No modification made to the opacity of thin tetrahedra.

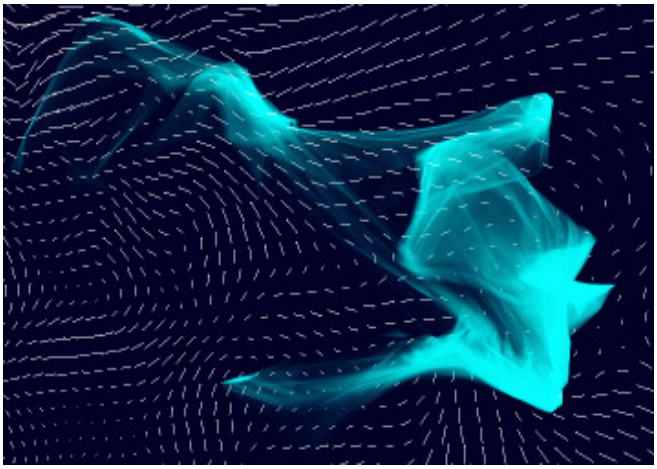

Figure 5b. Opacity of unusually thin tetrahedra is reduced.

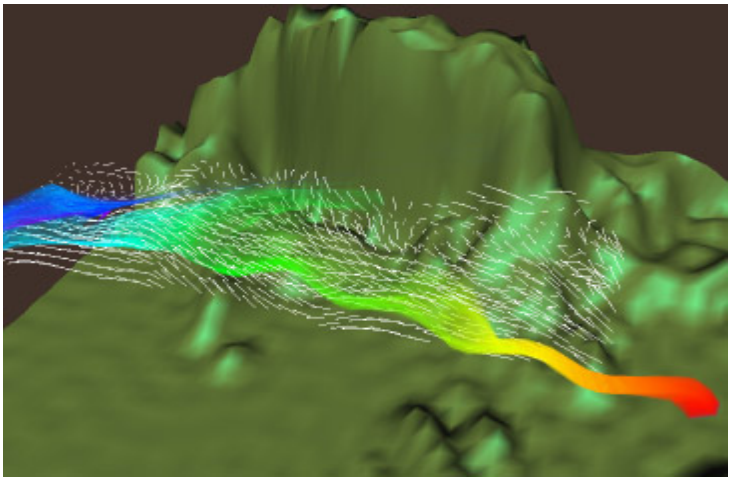

Figure 9. Dynamic flow volume colored by age in a curvilinear grid depicting simulated winds in Indonesia.

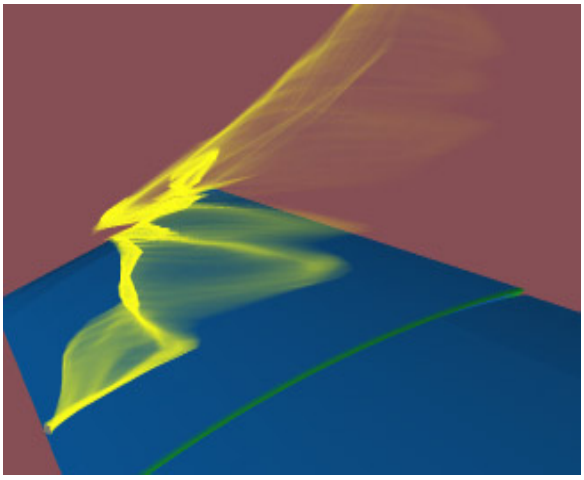

Figure 10. Frame from an animation of unsteady flow on a clipped delta wingan with oscillating flap. Data set has seven curvilinear grids.